Principais Comandos Git		
Comando	**Quando Utilizar**	**Pontos de Atenção**
git init	Inicializar um novo repositório Git na pasta atual	Certifique-se de estar no diretório correto antes de executar
git clone "<url>"	Clonar um repositório remoto	Verifique se você possui permissão para acessar o repositório
git add . ou git add <nome_do_arquivo>	Adicionar todos os arquivos ou arquivos específicos à área de stage	Use git add . com cuidado para não adicionar arquivos indesejados
git commit -m "descrição"	Criar um commit com uma mensagem descritiva	Certifique-se de que a mensagem do commit é clara e significativa
git commit -am "descrição"	Adicionar e commitar mudanças em um único comando	Funciona apenas para arquivos que já foram adicionados anteriormente
git commit --amend -m "nova descrição"	Modificar a mensagem do último commit	Use com cautela, pois reescreve o histórico do commit
git branch -M main	Renomear a branch atual para main	Útil para configurar o padrão moderno de branch principal
git remote add origin "<link_do_repositório>"	Conectar o repositório local ao remoto	Certifique-se de que o link está correto e você tem permissão de acesso
git push -u origin main	Enviar a branch main ao repositório remoto e configurar upstream	Configura main como a branch padrão para futuros push e pull
git push	Enviar commits locais ao repositório remoto	Verifique se você está na branch correta
git pull	Atualizar o repositório local com as mudanças do repositório remoto	Pode causar conflitos; revise antes de confirmar
git fetch	Buscar mudanças do repositório remoto sem mesclar	Use antes de git merge para revisar mudanças
git merge <nomeDaBranch>	Mesclar uma branch com a branch atual	Resolva conflitos de merge conforme necessário
git checkout -b "nomeDaNovaBranch"	Criar e mudar para uma nova branch	Use nomes descritivos para as branches
git checkout <nomeDaBranch>	Mudar para uma branch existente	Verifique o estado atual dos arquivos antes de mudar
git log	Exibir histórico de commits	Útil para revisar o histórico do projeto
git status	Verificar o estado dos arquivos no repositório	Use frequentemente para manter o controle do estado dos arquivos
git stash	Salvar temporariamente mudanças não commitadas	As mudanças salvas são removidas do workspace
git stash list	Listar stashes salvos	Acompanhe o que foi stashado
git stash pop	Aplicar e remover o stash mais recente	Use git stash list para verificar o que será aplicado
git stash apply	Aplicar um stash específico sem removê-lo da lista	Use para preservar o stash original
git restore <nome_do_arquivo>	Restaurar um arquivo ao estado do último commit	Pode ser usado para desfazer mudanças locais em um arquivo específico
git reflog	Exibir histórico de referências dos commits	Útil para recuperação de commits perdidos
git reset --soft <hash>	Redefinir HEAD para um commit específico, mantendo mudanças no stage	Use para desfazer commits sem perder alterações
git reset --mixed <hash>	Redefinir HEAD e o stage para um commit específico, mantendo mudanças no workspace	É o modo padrão do git reset
git reset --hard <hash>	Redefinir HEAD, stage e workspace para um commit específico	Todas as mudanças locais serão perdidas
git push --set-upstream origin <branch>	Configurar uma branch local para rastrear uma branch remota	Útil para novas branches
git fetch origin <branch>	Buscar uma branch específica do repositório remoto	Use para obter atualizações de uma branch específica sem mesclar

git branch	Listar branches locais	Verifique em qual branch você está atualmente
git branch -v	Listar branches com o commit mais recente	Útil para revisar rapidamente o estado das branches
git branch -vv	Listar branches com o commit mais recente e o nome da branch upstream	Ajuda a entender o estado de sincronização das branches
git branch -d <nomeBranch-Excluir> ou git branch -d "nomeBranch-Excluir"	Excluir uma branch local	Use aspas se o nome da branch contiver espaços ou caracteres especiais
git diff <branch1> <branch2>	Exibir diferenças entre duas branches	Útil para revisar mudanças antes de mesclar

Padronização de Commit		
Tipo de Commit	**Descrição**	**Exemplo**
feat	Introdução de uma nova funcionalidade	feat: adicionar página de login
fix	Correção de um bug	fix: corrigir bug no formulário de cadastro
docs	Alterações na documentação	docs: atualizar README com instruções de setup
style	Alterações que não afetam a lógica do código (espaços, formatação, ponto e vírgula, etc.)	style: formatar código no arquivo main.js
refactor	Alterações no código que não corrigem bugs nem adicionam funcionalidades	refactor: melhorar a estrutura do componente Header
perf	Alterações no código que melhoram o desempenho	perf: otimizar carregamento da página inicial
test	Adição ou correção de testes	test: adicionar testes para o componente Button
build	Alterações que afetam o sistema de build ou dependências externas	build: atualizar dependências no package.json
ci	Alterações em arquivos de configuração e scripts de CI (Continuous Integration)	ci: adicionar workflow para CI com GitHub Actions
chore	Outras alterações que não modificam o código de produção ou testes	chore: remover arquivos desnecessários
revert	Reversão de um commit anterio	revert: reverter commit abc123

Regras Gerais para Mensagens de Commit

- **Mensagem Curta e Descritiva:** A mensagem deve ser curta e descrever claramente a mudança.
  - Exemplo: feat: adicionar validação ao formulário de login
- **Mensagem no Imperativo**: Escreva a mensagem no imperativo, como se estivesse dando um comando.
  - **Correto:** feat: adicionar validação ao formulário
  - **Incorreto:** feat: adicionando validação ao formulário
- **Separar Assunto e Corpo com Linha em Branco:** Caso precise adicionar mais detalhes, separe a linha do assunto e o corpo da mensagem com uma linha em branco.
  - feat: adicionar validação ao formulário de login

    - Adiciona validação para campos de email e senha
    - Mostra mensagem de erro em caso de campos inválidos
- **Referenciar Issues ou Pull Requests:** Se o commit estiver relacionado a uma issue ou pull request, inclua a referência no corpo da mensagem.
  - fix: corrigir bug no formulário de cadastro

    - Corrige o bug onde o formulário não era enviado corretamente (#42).